Improving Feature Tracking with Robust Statistics

Andrea Fusiello[a†], Emanuele Trucco[b], Tiziano Tommasini[a], Vito Roberto[a]

[a] Dipartimento di Matematica e Informatica
Università di Udine,
Via delle Scienze 206, I-33100 Udine, IT
E-mail: {fusiello,tommasin,roberto}@dimi.uniud.it

[b] Department of Computing and Electrical Engineering
Heriot-Watt University
Riccarton, Edinburgh, EH14 4AS, UK
E-mail: mtc@cee.hw.ac.uk

[†] Corresponding author

Abstract

This paper addresses robust feature tracking. We extend the well-known Shi-Tomasi-Kanade tracker by introducing an *automatic* scheme for rejecting spurious features. We employ a simple and efficient outlier rejection rule, called X84, and prove that its theoretical assumptions are satisfied in the feature tracking scenario. Experiments with real and synthetic images confirm that our algorithm makes good features track better; we show a quantitative example of the benefits introduced by the algorithm for the case of fundamental matrix estimation. The complete code of the robust tracker is available via ftp.

Key words: Motion Analysis, Feature Tracking, Robust Statistics, Optical flow, Registration, X84

1 Introduction

Much work on structure from motion [12] has assumed that correspondences through a sequence of images could be recovered. Feature tracking finds matches by selecting image features and tracks these as they move from frame to frame. It can be seen as an instance of the general problem of computing the *optical flow*, that is, the vector's field that describes how the image is changing with time, at relatively sparse image positions [15, 2, 5]. The methods based on the detection of two dimensional features (such as corners) have the advantage that the full optical flow is known at every measurement position, because they do not suffer from the aperture problem effect (a discussion on this subject can be found in [24]). Works on tracking of two dimensional features include [13, 1, 6, 18, 26].

Robust tracking means detecting automatically unreliable matches, or *outliers*, over an image sequence (see [14] for a survey of robust methods in computer vision). Recent examples of such robust algorithms include [23], which identifies tracking outliers while estimating the fundamental matrix, and [22], which adopts a RANSAC [8] approach to eliminate outliers for estimating the trifocal tensor. Such approaches increase the computational cost of tracking significantly, as they are based on iterative algorithms.

This paper concentrates on the well-known Shi-Tomasi-Kanade tracker, and proposes a robust version based on an efficient outlier rejection scheme. Building on results from [13], Tomasi and Kanade [20] introduced a feature tracker based on SSD matching and assuming translational frame-to-frame displacements. Subsequently, Shi and Tomasi [19] proposed an *affine model*, which proved adequate for region matching over longer time spans. Their system classified a tracked feature as *good* (reliable) or *bad* (unreliable) according to the residual of the match between the associated image region in the first and current frames; if the residual exceeded a user-defined threshold, the feature was rejected. Visual inspection of results demonstrated good discrimination between good and bad features, but the authors did not specify how to reject bad features *automatically*.

This is the problem that our method solves. We extend the Shi-Tomasi-Kanade tracker (Section 2) by introducing an *automatic* scheme for rejecting spurious features. We employ a simple, efficient, model-free outlier rejection rule, called *X84*, and prove that its assumptions are satisfied in the feature tracking scenario (Section 3). Our RobustTracking algorithm is summarized in Section 4. Experiments with real and synthetic images confirm that our algorithm makes good features to track better, in the sense that outliers are located reliably (Section 5). We illustrate quantitatively the benefits introduced by the algorithm with the example of fundamental matrix estimation. Image sequences with results and the source code of the robust tracker are available on line (http://www.dimi.uniud.it/~fusiello/demo-rtr/).

2 The Shi-Tomasi-Kanade tracker

In this section the Shi-Tomasi-Kanade tracker [19, 20] will be briefly described. Consider an image sequence $I(\mathbf{x}, t)$, where $\mathbf{x} = [u, v]^\top$ are the coordinates of an image point. If the time sampling frequency (that is, the frame rate) is sufficiently high, we can assume that small image regions undergo a geometric transformation, but their intensities remain unchanged:

$$I(\mathbf{x}, t) = I(\delta(\mathbf{x}), t + \tau), \tag{1}$$

where $\delta(\cdot)$ is the *motion field*, specifying the *warping* that is applied to image points. The fast-sampling hypothesis allows us to approximate the motion with a translation, that is,

$$\delta(\mathbf{x}) = \mathbf{x} + \mathbf{d}, \tag{2}$$

where $\mathbf{d}$ is a displacement vector. The tracker's task is to compute $\mathbf{d}$ for a number of automatically selected point features for each pair of successive frames in the sequence. As the image motion model is not perfect, and because of image noise,

(1) is not satisfied exactly. The problem is then finding the displacement $\mathbf{d}$ which minimizes the SSD residual

$$\epsilon = \sum_W \left[I(\mathbf{x} + \mathbf{d}, t + \tau) - I(\mathbf{x}, t) \right]^2, \tag{3}$$

where W is a given feature window centered on the point $\mathbf{x}$. In the following we will solve this problem by means of a Newton-Raphson iterative search.
Thanks to the fast-sampling assumption, we can approximate $I(\mathbf{x} + \mathbf{d}, t + \tau)$ with its first-order Taylor expansion:

$$I(\mathbf{x}+\mathbf{d}, t+\tau) \approx I(\mathbf{x}, t) + \nabla I(\mathbf{x}, t)^\top \mathbf{d} + I_t(\mathbf{x}, t)\tau, \tag{4}$$

where $\nabla I^\top = [I_u, I_v] = [\partial I/\partial u, \partial I/\partial v]$ and $I_t = \partial I/\partial t$. We can then rewrite the residual (3) as

$$\epsilon \approx \sum_W (\nabla I(\mathbf{x}, t)^\top \mathbf{d} + I_t(\mathbf{x}, t)\tau)^2. \tag{5}$$

To minimize the residual (5), we differentiate it with respect to the unknown displacement $\mathbf{d}$ and set the result to zero, obtaining the linear system:

$$\mathbf{C}\mathbf{d} = \mathbf{g}, \tag{6}$$

where

$$\mathbf{C} = \sum_W \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix} \tag{7}$$

$$\mathbf{g} = -\tau \sum_W I_t \begin{bmatrix} I_u & I_v \end{bmatrix}^\top. \tag{8}$$

If $\mathbf{d}_k = \mathbf{C}^{-1}\mathbf{g}$ is the displacement estimate at iteration k, and assuming a unit time

interval between frames, the algorithm for minimizing (5) is the following:

$$\begin{cases} \mathbf{d}_0 = \mathbf{0} \\ \mathbf{d}_{k+1} = \mathbf{d}_k + \mathbf{C}^{-1} \sum\limits_{W} \Big[\left(I(\mathbf{x}, t) - I(\mathbf{x} + \mathbf{d}_k, t+1) \right) \nabla I(\mathbf{x}, t) \Big] \end{cases} .$$

2.1 Feature extraction

A feature is defined as a region that can be easily tracked from one frame to the other. In this framework, a *feature* can be tracked reliably if a numerically stable solution to (6) can be found, which requires that $\mathbf{C}$ is well-conditioned and its entries are well above the noise level. In practice, since the larger eigenvalue is bound by the maximum allowable pixel value, the requirement is that the smaller eigenvalue must be sufficiently large. Calling λ_1 and λ_2 the eigenvalues of $\mathbf{C}$, we accept the corresponding feature if

$$\min(\lambda_1, \lambda_2) > \lambda_t \tag{9}$$

where λ_t is a user-defined threshold [19].

Figure 1 near here

This algebraic characterization of "trackable" features has an interesting interpretation, as they turns out to be *corners*, that is image features characterized by an intensity discontinuity in two directions. Since the motion of an image feature can be measured only in its projection on the brightness gradient (*aperture problem*), corners are the features whose motion can be measured.

Discontinuity can be detected, for instance, using normalized cross-correlation, which measures how well an image patch matches other portions of the image as it is shifted from its original location. A patch which has a well-defined peak in its auto-correlation function can be classified as a corner. Let us compute the change in intensity, as the sum of squared differences, in the direction $\mathbf{h}$ for a patch W centered in $\mathbf{x} = (u, v)$:

$$E_{\mathrm{h}}(\mathbf{x}) = \sum_{\mathbf{d} \in W} \left(I(\mathbf{x} + \mathbf{d}) - I(\mathbf{x} + \mathbf{d} + \mathbf{h}) \right)^2 \tag{10}$$

Using the Taylor series expansion truncated to the linear term:

$$\begin{aligned} E_{\mathrm{h}}(\mathbf{x}) &\approx \sum_{\mathbf{d} \in W} \left(\nabla I(\mathbf{x} + \mathbf{d})^\top \mathbf{h} \right)^2 \\ &= \sum_{\mathbf{d} \in W} \mathbf{h}^\top (\nabla I(\mathbf{x} + \mathbf{d}))(\nabla I(\mathbf{x} + \mathbf{d}))^\top \mathbf{h} \\ &= \sum_{\mathbf{d} \in W} \mathbf{h}^\top \begin{pmatrix} I_u^2 & I_u\, I_v \\ I_u\, I_v & I_v^2 \end{pmatrix} \mathbf{h} \\ &= \mathbf{h}^\top \left(\sum_{\mathbf{d} \in W} \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix} \right) \mathbf{h}. \end{aligned} \tag{11}$$

The change in intensity around $\mathbf{x}$ is therefore given by

$$E_{\mathrm{h}}(\mathbf{x}) = \mathbf{h}^\top \mathbf{C}\ \mathbf{h} \tag{12}$$

where $\mathbf{C}$ is just the matrix defined in (7). Elementary eigenvector theory tells us that, since $||\mathbf{h}|| = 1$, then

$$\lambda_1 < E_{\mathrm{h}}(\mathbf{x}) < \lambda_2, \tag{13}$$

where λ_1 and λ_2 are the eigenvalues of $\mathbf{C}$. So, if we try every possible orientation $\mathbf{h}$, the maximum change in intensity we will find is λ_2, and the minimum value is λ_1. We can therefore classify the structure around each pixel by looking at the eigenvalues of $\mathbf{C}$:

- no structure: $\lambda_1 \approx \lambda_2 \approx 0$;
- edge: $\lambda_1 \approx 0$, $\lambda_2 \gg 0$;
- corner: λ_1 e λ_2 both large and distinct.

Hence, the features selected according to criterion criterion (9) are to be interpreted as corners. Indeed, this method is very closely related to some classical corner detectors, such as [16, 17, 11].
Figure 1 shows the value of the minimum eigenvalue for the first frame of the "Artichoke" sequence (see Section 5).

2.2 Affine model

The translational model cannot account for certain transformations of the feature window, for instance rotation, scaling, and shear. An *affine motion field* is a more accurate model [19], that is,

$$\delta(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{d}, \tag{14}$$

where $\mathbf{d}$ is the displacement, and $\mathbf{M}$ is a 2×2 matrix accounting for *affine warping*, and can be written as $\mathbf{M} = \mathbf{1} + \mathbf{D}$, with $\mathbf{D} = [d_{ij}]$ a deformation matrix and $\mathbf{1}$ the identity matrix. Similarly to the translational case, one estimates the motion parameters, $\mathbf{D}$ and $\mathbf{d}$, by minimizing the residual

$$\epsilon = \sum_W \big[I(\mathbf{M}\mathbf{x} + \mathbf{d}, t + \tau) - I(\mathbf{x}, t)\big]^2. \tag{15}$$

By plugging the first-order Taylor expansion of $I(\mathbf{M}\mathbf{x} + \mathbf{d}, t + \tau)$ into (15), and imposing that the derivatives with respect to $\mathbf{D}$ and $\mathbf{d}$ are zero, we obtain the linear system

$$\mathbf{B}\mathbf{z} = \mathbf{f}, \tag{16}$$

in which $\mathbf{z} = [d_{11}\ d_{12}\ d_{21}\ d_{22}\ d_1\ d_2]^\top$ contains the unknown motion parameters, and

$$\mathbf{f} = -\tau \sum_W I_t \left[uI_u \; uI_v \; vI_u \; vI_v \; I_u \; I_v \right]^\top ,$$

with

$$\mathbf{B} = \sum_W \begin{bmatrix} \mathbf{U} & \mathbf{V} \\ \mathbf{V}^\top & \mathbf{C} \end{bmatrix} ,$$

$$\mathbf{U} = \begin{bmatrix} u^2 I_u^2 & u^2 I_u I_v & uv I_u^2 & uv I_u I_v \\ u^2 I_u I_v & u^2 I_v^2 & uv I_u I_v & uv I_v^2 \\ uv I_u^2 & uv I_u I_v & v^2 I_u^2 & v^2 I_u I_v \\ uv I_u I_v & uv I_v^2 & v^2 I_u I_v & v^2 I_v^2 \end{bmatrix} ,$$

$$\mathbf{V}^\top = \begin{bmatrix} uI_u^2 & uI_u I_v & vI_u^2 & vI_u I_v \\ uI_u I_v & uI_v^2 & vI_u I_v & vI_v^2 \end{bmatrix} .$$

Again, (15) is solved for $\mathbf{z}$ using a Newton-Raphson iterative scheme.

If frame-to-frame affine deformations are negligible, the pure translation model is preferable (the matrix $\mathbf{M}$ is assumed to be the identity). The affine model is used for comparing features between frames separated by significant time intervals to monitor the quality of tracking.

3 Robust monitoring

In order to monitor the quality of the features tracked, the tracker checks the residuals between the first and the current frame: high residuals indicate bad features which must be rejected. Following [19], we adopt the affine model, as a pure translational model would not work well with long sequences: too many good features

are likely to undergo significant rotation, scaling or shearing, and would be incorrectly discarded. Non-affine warping, which will yield high residuals, is caused by *occlusions*, perspective distortions and strong intensity changes (e.g. specular reflections). This section introduces our method for selecting a robust rejection threshold *automatically*.

3.1 Distribution of the residuals

We begin by establishing which distribution is to be expected for the residuals when comparing good features, i.e., almost identical regions. We assume that the intensity $I(\delta(\mathbf{x}), t)$ of each pixel in the current-frame region is equal to the intensity of the corresponding pixel in the first frame $I(\mathbf{x}, 0)$ plus some Gaussian noise $n \equiv \eta(0, 1)$[†]. Hence

$$I(\delta(\mathbf{x}), t) - I(\mathbf{x}, 0) \equiv \eta(0, 1).$$

Since the square of a Gaussian random variable has a chi-square distribution, we obtain

$$\left[I(\delta(\mathbf{x}), t) - I(\mathbf{x}, 0)\right]^2 \equiv \chi^2(1).$$

The sum of n chi-square random variables with one degree of freedom is distributed as a chi-square with n degrees of freedom (as it is easy to see by considering the moment-generating functions). Therefore, the residual computed according to (3) over a $N \times N$ window W is distributed as a chi-square with N^2 degrees of freedom:

$$\epsilon = \sum_W \left[I(\delta(\mathbf{x}), t) - I(\mathbf{x}, 0)\right]^2 \equiv \chi^2(N^2). \tag{17}$$

Figure 2 near here

[†] $\equiv$ means that the variable to the left has the probability distribution specified to the right.

As the number of degrees of freedom increases, the chi-square distribution approaches a Gaussian, which is in fact used to approximate the chi-square with more than 30 degrees of freedom. Therefore, since the window W associated to each feature is at least 7×7, we can safely assume a Gaussian distribution of the residual for the good features:

$$\epsilon \equiv \eta(N^2, 2N^2).$$

3.2 The X84 rejection rule

When the two regions over which we compute the residual are bad features (that is, they are not warped by an affine transformation), the residual is not a sample from the Gaussian distribution of good features: it is an *outlier*. Hence, the detection of bad features reduces to a problem of outlier detection. This is equivalent to the problem of estimating the mean and variance of the underlying Gaussian distribution from the corrupted data ϵ_i, the residuals (given by (3)) between the i-th feature in the last frame and the same feature in the first frame. To do this, we employ a simple but effective model-free rejection rule, *X84* [10], which use robust estimates for location and scale to set a rejection threshold. The median is a robust location estimator, and the Median Absolute Deviation (MAD), defined as

$$\text{MAD} = \underset{i}{\text{med}}\{|\epsilon_i - \underset{j}{\text{med}}\ \epsilon_j|\}. \tag{18}$$

is a robust estimator of the scale (i.e., the spread of the distribution). It can be seen that, for symmetric (and moderately skewed) distributions, the MAD coincides with the *interquartile range*:

$$\text{MAD} = \frac{\xi_{3/4} - \xi_{1/4}}{2}, \tag{19}$$

where ξ_q is the qth quantile of the distribution (for example, the median is $\xi_{1/2}$). For normal distributions we infer the standard deviation from

$$\mathrm{MAD} = \Phi^{-1}(3/4)\sigma \approx 0.6745\sigma. \tag{20}$$

The X84 rule prescribes to reject values that are more than k Median Absolute Deviations away from the median. A value of k=5.2, under the hypothesis of Gaussian distribution, is adequate in practice, as it corresponds to about 3.5 standard deviations, and the range $[\mu - 3.5\sigma, \mu + 3.5\sigma]$ contains more than the 99.9% of a Gaussian distribution . The rejection rule X84 has a *breakdown point* of 50%: any majority of the data can overrule any minority.

3.3 Photometric normalization

Our robust implementation of the Shi-Tomasi-Kanade tracker incorporates also a *normalized* SSD matcher for residual computation. This limits the effects of intensity changes between frames, by subtracting the average grey level (μ_J, μ_I) and dividing by the standard deviation (σ_J, σ_I) in each of the two regions considered:

$$\epsilon = \sum_W \left[\frac{J(\mathbf{Mx} + \mathbf{d}) - \mu_J}{\sigma_J} - \frac{I(\mathbf{x}) - \mu_I}{\sigma_I} \right]^2, \tag{21}$$

where $J(\cdot)$=$I(\cdot, t+1)$, $I(\cdot)$=$I(\cdot, t)$.
It can be easily seen that this normalization is sufficient to compensate for intensity changes modeled by $J(\mathbf{Mx} + \mathbf{d}) = \alpha I(\mathbf{x}) + \beta$. A more elaborate normalization is described in [7], whereas [9] reports a modification of the Shi-Tomasi-Kanade tracker based on explicit photometric models.

4 Summary of the RobustTracking algorithm

The RobustTracking algorithm can be summarized as follows:

1. given an image sequence;
2. filter the sequence with a Gaussian kernel in space and time (for the selection of the scale of the kernel, see [4]);
3. select features to be tracked according to (9);
4. register features in each pair of consecutive frames in the sequence, using translational warping (2);
5. in the last frame of the sequence, compute the residuals between this and the first frame, for each feature, using affine warping (14);
6. reject outlier features according to the X84 rule (9).

The decision of which frame is deemed to be the *last* one is left open; the only, obvious, constraint is that a certain fraction of the features present in the first frame should be still visible in the last. On the other hand, monitoring cannot be done at every frame, because the affine warping would not be appreciable.

5 Experimental results

We evaluated our tracker in a series of experiments, of which we report the most significant ones.

"Platform" (Figure 3, 256×256 pixels). A 20-frame synthetic sequence, simulating a camera rotating in space while observing a subsea platform sitting on the seabed (real seabed acquired by a sidescan sonar, rendered as an intensity image, and texture-mapped onto a plane). This sequence is part of the SOFA synthetic sequences (http://www.cee.hw.ac.uk/ mtc/sofa).

"Hotel" (Figure 4, 480×512 pixels). A static scene observed by a moving camera rotating and translating (59 frames). This is a well-known sequence from the CMU VASC Image Database (http://www.ius.cs.cmu.edu/idb/).

"Stairs" (Figure 5, 512×768 pixels). A 60-frame sequence of a white staircase sitting on a metal base and translating in space, acquired by a static camera. The base is the platform of a translation stage operated by a step-by-step motor under computer control (courtesy of F. Isgrò, Computer Vision Group, Heriot-Watt University).
"Artichoke" (Figure 6, 480×512 pixels). A 99-frame sequence taken with a camera translating in front of a static scene. This sequence can be found at the CMU VASC Image Database, and was used also by [21].

Figure 3 near here

Figure 4 near here

Figure 5 near here

Figure 6 near here

"Platform" is the only synthetic sequence shown here. No features become occluded, but notice the strong effects of the coarse spatial resolution on straight lines. We plotted the residuals of all features against the frame number (Figure 7). All features stay under the threshold computed automatically by X84, apart from one that is corrupted by the interference of the background. In "Stairs", some of the features picked up in the first frame are specular reflections from the metal platform, the intensity of which changes constantly during motion. The residuals for such features become therefore very high (Figure 9). All these features are rejected correctly. Only one good feature is dropped erroneously (the bottom left corner of the internal triangle), because of the strong intensity change of the inside of the block. In the "Hotel" sequence (Figure 8), all good features but one are preserved. The one incorrect rejection (bottom center, corner of right balcony) is due to the warping caused by the camera motion, too large to be accommodated by the affine model. The only spurious feature present (on the right-hand side of the stepped-house front) is rejected correctly. All features involved in occlusions in the "Artichoke" sequence (Figure 10) are identified and rejected correctly. Four good features out of 54 are also rejected (on the signpost on the right) owing to a marked contrast change in time between the pedestrian figure and the signpost in the background.
In our tests on a SPARCServer 10 running Solaris 2.5, the initial feature extraction

phase took 38s for "Platform" and 186s for "Artichoke", with a 15×15 window. The tracking phase took on average 1.6s per frame, independently from frame dimensions. As expected, extraction is very computationally demanding, since the eigenvalues of the $\mathbf{C}$ matrix are to be computed *for each pixel*. However, this process can implemented on a parallel architecture, thereby achieving real-time performances (30Hz), as reported in [3].

Figure 7 and 8 near here

Figure 9 and 10 near here

5.0.1 Quantifying improvement: an example

To illustrate quantitatively the benefits of our robust tracker, we used the feature tracked by robust and non-robust versions of the tracker to compute the fundamental matrix between the first and last frame of each sequence, then computed the RMS distance of the tracked points from the corresponding epipolar lines, using Zhang's implementation [25] of the *8-point algorithm*. If the epipolar geometry is estimated exactly, all points should lie on epipolar lines. The results are shown in Table 1. The robust tracker brings always a decrease in the RMS distance. Notice the limited decrease and high residual for "Platform"; this is due to the significant spatial quantization and smaller resolution, which worsens the accuracy of feature localization.

Table 1 near here

6 Conclusions

We have presented a robust extension of the Shi-Tomasi-Kanade tracker, based on the X84 outlier rejection rule. The computational cost is much less than that of schemes based on robust regression and random sampling like RANSAC or *Least Median of Squares* [14, 22], yet experiments indicate excellent reliability in the presence of non-affine feature warping (most right features preserved, all wrong features rejected). Our experiments have also pointed out the pronounced sensitivity of the

Shi-Tomasi-Kanade tracker to illumination changes. We believe that our robust tracker can be useful to the large community of researchers needing efficient and reliable trackers. To facilitate dissemination and enable direct comparisons and experimentation, we have made the code available on the Internet.

Acknowledgements

This work was supported by a British Council-MURST/CRUI grant and by the EU programme SOCRATES.

References

[1] S. T. Barnard and W. B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, 1980.

[2] J. L. Barron, D. J. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[3] A. Benedetti and P. Perona. Real-time 2-d feature detection on a reconfigurable computer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–593, Santa Barbara, CA, June 1998. IEEE Computer Society Press.

[4] J. W. Brandt. Improved accuracy in gradient-based optical flow estimation. *International Journal of Computer Vision*, 25(1):5–22, 1997.

[5] M. Campani and A. Verri. Motion analysis from first order properties of optical flow. *CVGIP: Image Understanding*, 56:90–107, 1992.

[6] D. Charnley and R. J. Blisset. Surface reconstruction from outdoor image sequences. *Image and Vision Computing*, 7(1):10–16, 1989.

[7] I.J. Cox, S. Roy, and S.L. Hingorani. Dynamic histogram warping of image pairs for constant image brightness. In *Proceedings of the IEEE International*

Conference on Image Processing, volume 2, pages 366–369, Washington, D.C, 1995.

[8] M. A. Fischler and R. C. Bolles. Random Sample Consensus: a paradigm model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[9] G.D. Hager and P.N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–410, San Francisco, CA, 1996.

[10] F.R. Hampel, P.J. Rousseeuw, E.M. Ronchetti, and W.A. Stahel. *Robust Statistics: the Approach Based on Influence Functions.* Wiley Series in probability and mathematical statistics. John Wiley & Sons, 1986.

[11] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 189–192, August 1988.

[12] T. S. Huang and A. N. Netravali. Motion and structure from feature correspondences: A review. *Proceedings of IEEE*, 82(2):252–267, 1994.

[13] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981.

[14] P. Meer, D. Mintz, D. Y. Kim, and A. Rosenfeld. Robust regression methods in computer vision: a review. *International Journal of Computer Vision*, 6:59–70, 1991.

[15] A. Mitchie and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *International Journal of Computer Vision*, 19(1):29–55, 1996.

[16] H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 584, August 1977.

[17] J.A. Noble. Finding corners. *Image and Vision Computing*, 6:121–128, May 1988.

[18] L.S. Shapiro, H. Wang, and J.M. Brady. A matching and tracking strategy for independently moving objects. In *Proceedings of the British Machine Vision Conference*, pages 306–315. BMVA Press, 1992.

[19] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

[20] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Pittsburg, PA, April 1991.

[21] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography – a factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[22] P. H. S. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. In R. Fisher and E. Trucco, editors, *British Machine Vision Conference*, pages 655–664. BMVA, September 1996. Edinburgh.

[23] P. H. S. Torr, A. Zisserman, and S. Maybank. Robust detection of degeneracy. In E. Grimson, editor, *Proceedings of the IEEE International Conference on Computer Vision*, pages 1037–1044. Springer–Verlag, 1995.

[24] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice-Hall, 1998.

[25] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. Technical Report 2927, INRIA Sophia-Antipolis, France, July 1996.

[26] Q. Zheng and R. Chellappa. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. *International Journal of Computer Vision*, 15(15):31–76, 1995.

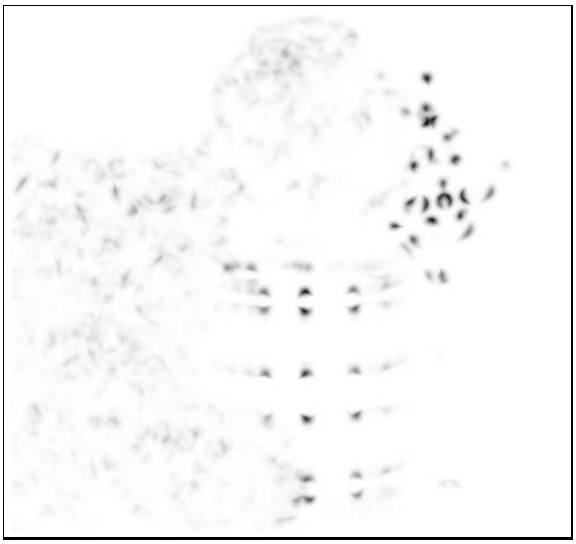

Figure 1: Value of $\min(\lambda_1, \lambda_2)$ for the first frame of 'Artichoke". Window size is 15 pixels. Darker points have an higher minimum eigenvalue.

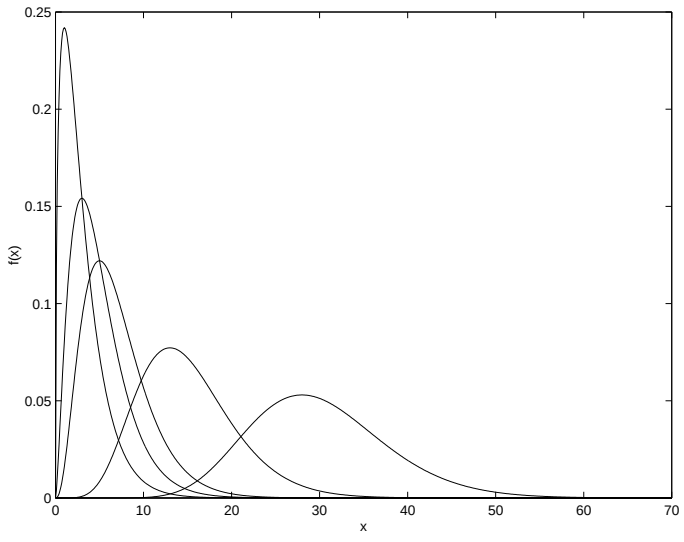

Figure 2: Chi-square density functions with 3,5,7,15 and 30 degrees of freedom (from left to right).

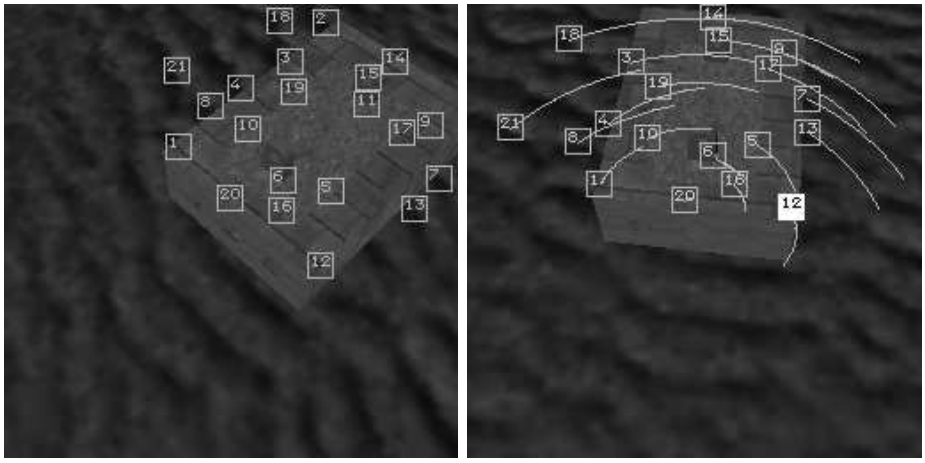

Figure 3: First (left) and last frame of the "Platform" sequence. In the last frame, filled windows indicate features rejected by the robust tracker.

Figure 4: First (left) and last frame of the "Hotel" sequence. In the last frame, filled windows indicate features rejected by the robust tracker.

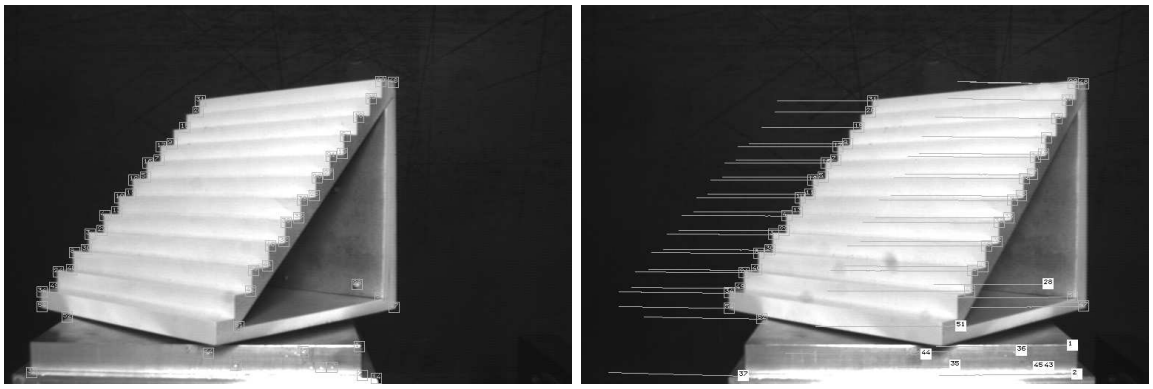

Figure 5: First (left) and last frame of the "Stairs" sequence. In the last frame, filled windows indicate features rejected by the robust tracker.

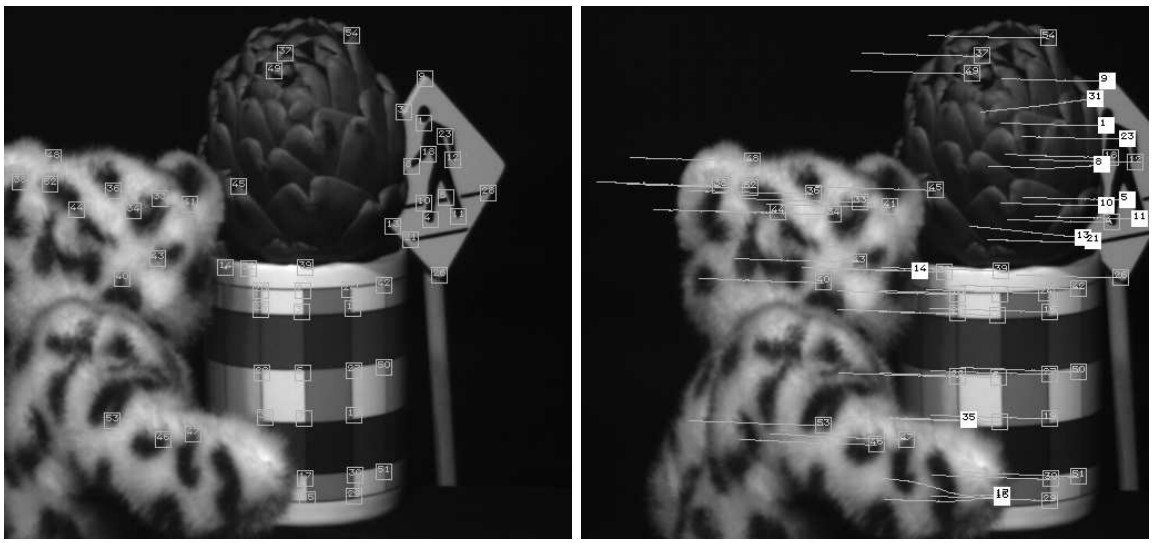

Figure 6: First (left) and last frame of the "Artichoke" sequence. In the last frame, filled windows indicate features rejected by the robust tracker.

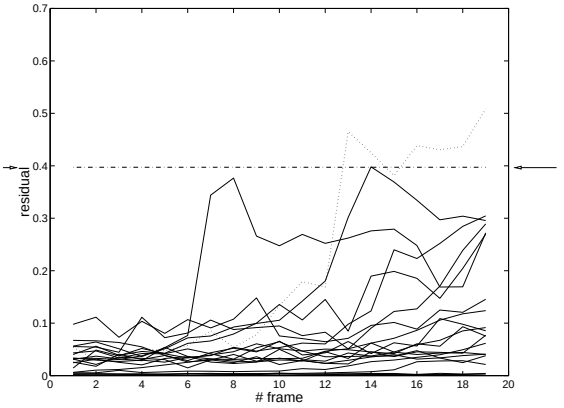

Figure 7: Residuals magnitude against frame number for "Platform". The arrows indicate the threshold set automatically by X84 (0.397189).

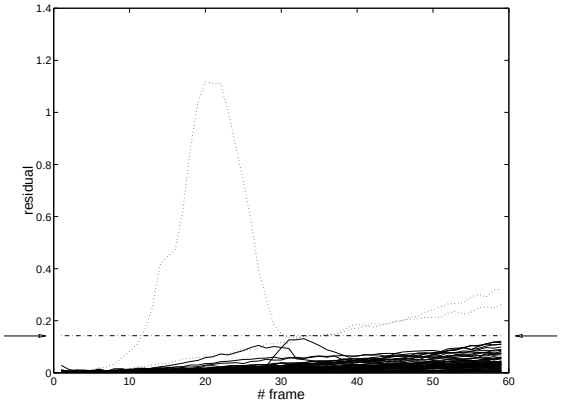

Figure 8: Residuals magnitude against frame number for "Hotel". The arrows indicate the threshold set automatically by X84 (0.142806).

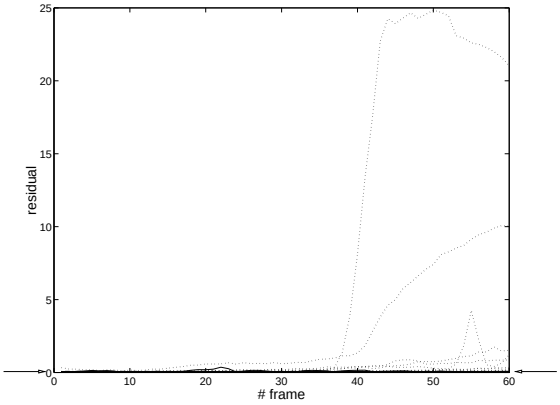

Figure 9: Residuals magnitude against frame number for "Stairs". The arrows indicate the threshold set automatically by X84 (0.081363) .

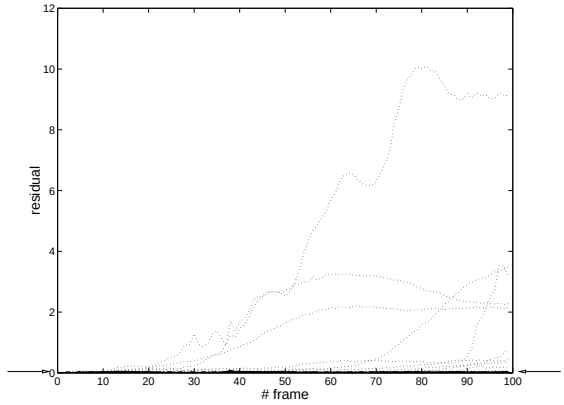

Figure 10: Residuals magnitude against frame number for "Artichoke". The arrows indicate the threshold set automatically by X84 (0.034511).

	Artichoke	Hotel	Stairs	Platform
All	1.40	0.59	0.66	1.49
X84	0.19	0.59	0.15	1.49

Table 1: RMS distance of points from epipolar lines. The first row gives the distance using all the features tracked (non-robust tracker), the second using only the features kept by X84 (robust tracker).